

✓ EXPERIMENT SEVEN

✓ Experiment 7: Min-Max Algorithm

Design a simple two-player game using the Min-Max algorithm in Python.

The game should simulate an adversarial environment where one player tries to maximize their score while the other tries to minimize it. Your task is to implement the core Min-Max algorithm, build a simple decision tree of game states, and determine the optimal move

```
# Min-Max Algorithm Implementation for a simple game (Tic-Tac-Toe or abstract tree)
```

```
# Example Game Tree (Leaf node values for MIN-MAX)
```


```
game_tree = {
    "root": ["A", "B", "C"],
    "A": ["A1", "A2", "A3"],
    "B": ["B1", "B2", "B3"],
    "C": ["C1", "C2", "C3"],
    "A1": 3, "A2": 5, "A3": 2,
    "B1": 9, "B2": 1, "B3": 6,
    "C1": 0, "C2": -1, "C3": 7
}
```



```
# Recursive Minimax function
```

```
def minimax(node, depth, is_maximizing):
    if isinstance(game_tree[node], int):
        return game_tree[node]

    if is_maximizing:
        best_val = float('-inf')
        for child in game_tree[node]:
            val = minimax(child, depth + 1, False)
            best_val = max(best_val, val)
        return best_val
    else:
        best_val = float('inf')
        for child in game_tree[node]:
            val = minimax(child, depth + 1, True)
            best_val = min(best_val, val)
        return best_val
```

```
# Start Min-Max evaluation
```

```
optimal_value = minimax("root", 0, True)
print(f" Optimal value for MAX player at root: {optimal_value}")
```

```
  Optimal value for MAX player at root: 2
```